Recuperación de Información: una Introducción

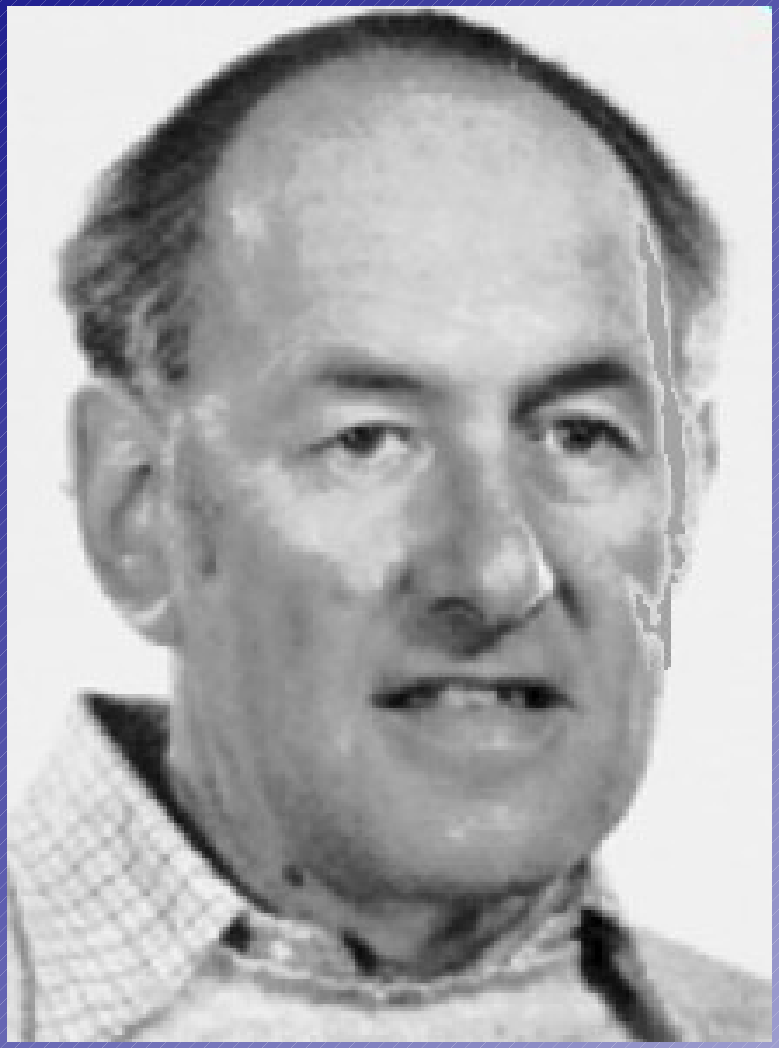
Alfonso E. Romero

Dep. Ciencias de la Computación e I.A.
Universidad de Granada
aeromero@decsai.ugr.es
http://decsai.ugr.es/~aeromero

E.P.S. Linares, 15 de Mayo de 2006

ÍNDICE

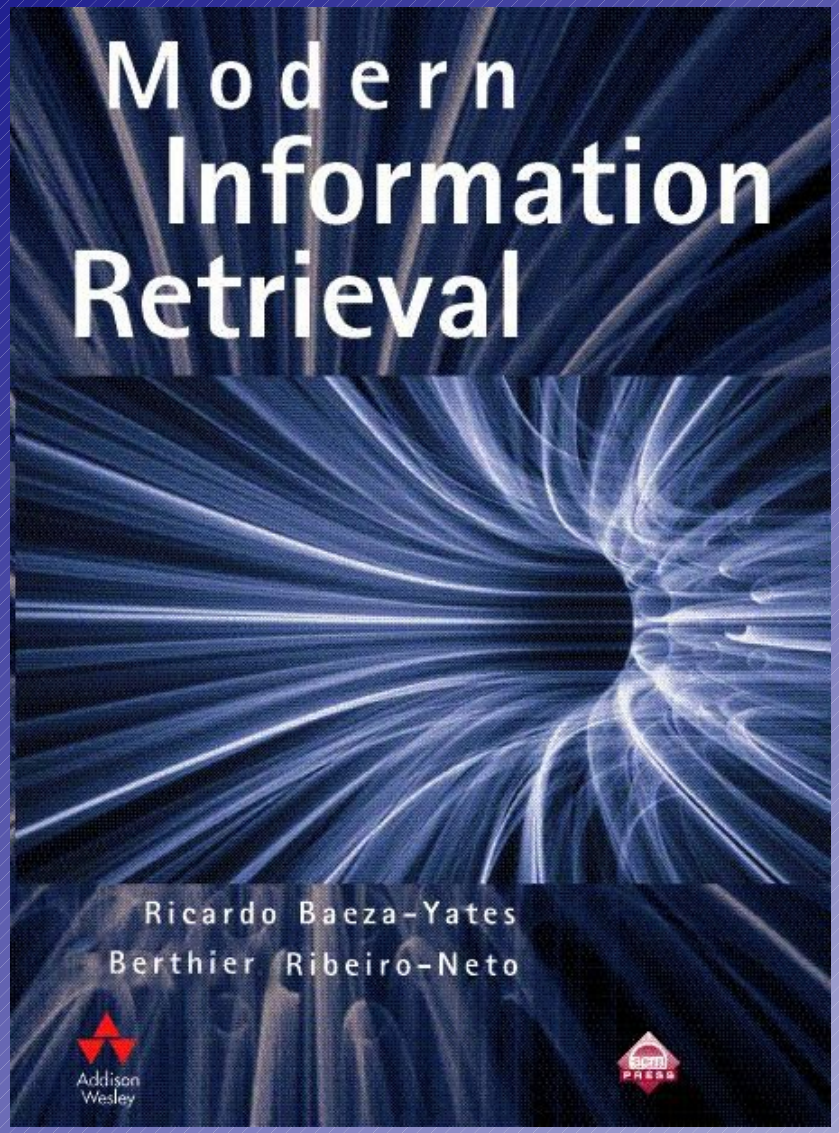
1.1 ¿Qué es la Recuperación de información (RI)?



G. Salton, padre de la R.I.

"Information retrieval is concerned with the representation, storage, organization, and accessing of information items"

1.2.- El documento: la unidad de información



'Document: a unit of retrieval. It might be a *paragraph*, a *section*, a *chapter*, a *Web page*, an *article*, or a *whole book*.' (Baeza-Yates & Ribeiro-Neto, 1999: 440)

- Documento como unidad de información
- No tiene por qué ser un documento de texto (imágenes, sonido, ... Siempre que se adapte la terminología)
- La información no se trata aquí con el sentido de Shannon (aunque la teoría de la Información tiene bastantes aplicaciones en RI), sino como un conjunto de datos que ya existen.

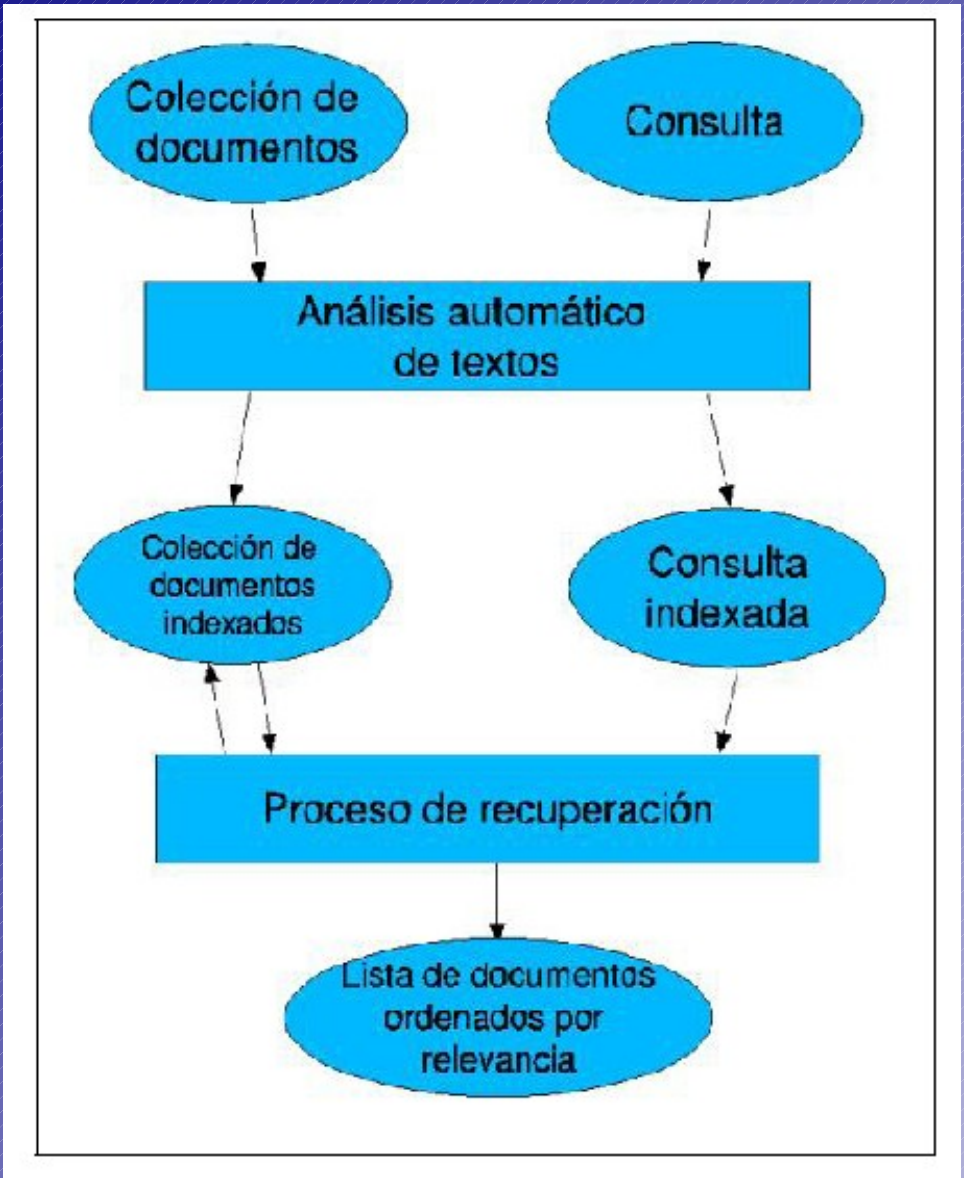
1.3.- Sistemas de R.I

El **objetivo** de los Sistemas de Recuperación de Información (SRI) es, dada una colección de documentos y una consulta formulada por un usuario en un cierto momento, proporcionar el subconjunto de documentos que es más relevante para la consulta del usuario.

Funcionamiento de un SRI:

1. El usuario introduce una *consulta* en el sistema. Esta consulta representa sus necesidades de información.
2. El sistema *procesa* dicha consulta. Se buscan documentos que, de alguna forma, sean *coincidentes* con los términos que aparecen en dicha consulta.
3. El sistema muestra los documentos que son coincidentes con la consulta, ordenándolos de mayor a menor relevancia según el valor proporcionado por una función de *ranking*.

1.3.- Sistemas de R.I (Gráficamente...)



Reorganizando conceptos:
¿Qué es necesario para hacer RI?

- Representación interna del documento (eficiente)

- Representación de la necesidad de información del usuario (consulta)

- Consultas, documentos.. ¿Representados de la misma forma? ¿Cómo representar relaciones?

- Función de ranking (medir la relevancia de un documento) ¿Cómo?

1.4.- Modelos de R.I.

Un Modelo de Recuperación de Información [Baeza]
es un cuádrupla $(D,Q,F,R(q_i,d_j))$, donde:

1. ***D*** es un conjunto formado por la representación (vistas lógicas) de los documentos.
2. ***Q*** es un conjunto formado por consultas, es decir la representación (vistas lógicas) de la información que el usuario necesita.
3. ***F*** es un marco o modelo de representación de los documentos, las consultas, y las relaciones existentes entre ellos.
4. $R(q_i,d_j)$ es una función (de clasificación o *ranking*) que asocia un número real a cada consulta q_i *de* Q y representación del documento d_j *de D*.

"La Recuperación de información es el estudio de diferentes Modelos de Recuperación de información, las técnicas necesarias para su implementación, la aplicabilidad de los mismos a problemas reales y su evaluación".

2.1.- Representación de los documentos

La *representación clásica* de documentos: “bag of words”.
Suponemos que los términos índice son *independientes* unos de otros (no importa el orden), por lo que, para cada par término t_i documento d_j tendemos un peso *w(i,j)* (dependiente del modelo).

De los documentos se eliminan palabras que no tienen un significado por sí mismas (stopwords) como preposiciones, pronombres, ...
Además, se realiza *case folding* y se eliminan caracteres no alfanuméricos

Los términos similares (con la misma raíz), son proyectados a la misma entidad (con objeto de no perder información si realizo una búsqueda con el término en plural). Esto se denomina *stemming.*

Se dispone de una estructuras (en disco) con los documentos procesados de esta forma, con los pesos precalculados y de fácil acceso. Las estructuras se crean una sola vez (*indexación*).

2.2.- El modelo booleano

- Trata el problema de recuperación de información como un problema de **teoría de conjuntos** y **álgebras de Boole**

- Los valores posibles de los pesos *w(i,j)* son **1** (si t_i aparece en d_j) ó **0**, en caso contrario.

- Los valores posibles de la función *ranking* son **1** (el documento es relevante) ó **0** (el documento no es relevante).

- Una consulta es una expresión booleana de términos. Por ejemplo: *(Fernando Y Alonso) O (Michael Y Schumacher Y NO(Ralph))*.

- Los documentos que se devuelven como relevantes son aquellos que verifican completamente dicha expresión booleana.

- Parecido a la búsqueda en bases de datos relacionales (“lo que el usuario ha expresado como necesario es lo que se devuelve”)

2.2.- El modelo booleano (2)

Análisis Crítico

 Modelo simple (fácil de implementar)

 Sólo considera acoplamientos exactos. Recupera mucho o poco, y lo que es peor (si es mucho) desordenado. En el ejemplo anterior, si estamos consultando documentos sólo de F1 vamos a tener un resultado enorme y desordenado.

 Dejamos demasiado trabajo al usuario: no todo el mundo conoce la teoría básica de conjuntos...

 No considera más importante un documento en el que la consulta sea más frecuente que otro (términos igualmente importantes).

2.3.- El modelo de espacio vectorial

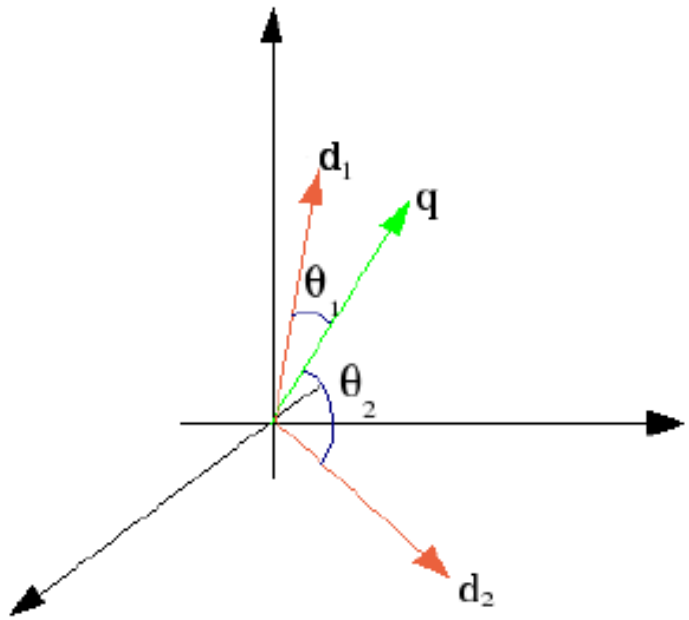
- Salton y McGill en el sistema SMART (1971)

- Sea una aplicación *f : T* --> *N* (donde *T* es el conjunto de términos). Si consideramos cada término como una *dimensión* en un espacio vectorial, podemos **representar cada documento como un vector** donde en cada dimensión *i* colocamos un escalar que represente la importancia del término *i* en ese documento.

- Si representamos las consultas también mediante un vector (con un 1.0 para cada uno de los términos que aparezcan), calcular la similaridad entre la consulta y cada uno de los documentos puede plantearse como calcular **el ángulo entre el vector consulta y cada uno de los vectores de los documentos.**

- Los documentos más “paralelos” a la consulta serán aquellos que respondan mejor a la misma.

2.3.- El modelo de espacio vectorial (2)



- En el gráfico, d_1 "responde" mejor a la consulta que d_2.
- Las coordenadas con valor distinto de 0 para el vector consulta son muy pocas: **nótese que sólo tenemos que realizar la operaciones para ellas (supondremos que si un término no aparece en un documento su coordenada es 0).**
- La forma de calcular la relevancia está expresada aquí (ángulo entre dos vectores):

$$R(q,d_j) = \cos(\widehat{q,d_j}) = \frac{\vec{q}\cdot\vec{d}_j}{|q|\cdot|d_j|} = \frac{\sum_{k=1}^{T} w_{k,q}\cdot w_{k,j}}{\sqrt{\sum_{k=1}^{T}(w_{k,q})^2\cdot\sum_{k=1}^{T}(w_{k,j})^2}}$$

2.3.- El modelo de espacio vectorial (3)

Problema: **¿cómo calcular los pesos w(i,j)?** (o, de otra forma, **¿cómo medir la importancia de un término en un documento?**).

- A **mayor frecuencia** de un término en un documento, **mayor importancia** en éste... ¿Cómo normalizar la frecuencia para evitar que se dispare la importancia de un término en documentos muy grandes comparados con documentos pequeños? Calculamos el $\mathrm{tf} = \frac{n_i}{\sum_k n_k}$, con lo que todas las frecuencias estarán entre 0 y 1, para cada documento.

- Otro punto a tener en cuenta es que no todos los términos son igual de importantes dentro de la misma colección. Responder muy bien a muchos términos infrecuentes en la consulta y no muy bien a pocos frecuentes es mejor que hacerlo al revés. Los términos que aparecen en muchos documentos **no discriminan nada.** Una medida de la rareza del término es la *frecuencia inversa documental* (ó *idf*), definida como $idf_i = \log \frac{N}{n_i}$

- Combinando ambas: $w(i,j) = tf_{i,j} \cdot idf_i$ (esquema TF x IDF).

2.3.- El modelo de espacio vectorial (4)

Análisis Crítico

 Recuperación “ordenada” (si un documento se devuelve antes que otro, es más relevante).

 Acoplamiento “parcial” (el documento más relevante no tiene por qué contener todos los términos de la consulta). Incluso, no tendria por qué haber ningún documento así, aunque hubiera documentos relevantes.

 Comparado con otros intentos de recuperación ordenada funciona bastante bien (difícil de superar).

 El esquema TF x IDF es intuitivo (pero no formal)

 Asume independencia de los términos (*bag of words*).

2.4.- El modelo probabilístico básico

- Propuesto por K. Spack-Jones y S. Robertson (sistema OKAPI,1976)
- Supone los **pesos binarios** (en {0, 1}, si no aparece un término en el documento, o sí lo hace, respectivamente).
- Utiliza un marco basado en la **teoría de la probabilidad** (hoy día hay muchas aproximaciones con dicha teoría, de ahí el nombre).
- **Definición:** sea *R* el conjunto de documentos relevantes para una cierta consulta, y sea ¬*R* su complementario (dicho conjunto es desconocido).
- Sea $P(R|d_j)$ la probabilidad de que el documento d_j sea relevante, y $P(\neg R|d_j)$ la probabilidad de que dicho documento no sea relevante.

El valor de relevancia de un documento dada una consulta lo definimos como el cociente:

$$R(q, dj) = P(R|d_j) / P(\neg R|d_j)$$

2.4.- El modelo probabilístico básico (2)

- Aplicando el T. Bayes, obtenemos

$$\mathrm{R}(q,d_j) = \frac{P(\vec{d}_j|R)\,P(R)}{P(\vec{d}_j|\bar{R})\,P(\bar{R})}$$

- Como, P(R) y P(¬R) son constantes, los podemos eliminar del cociente.

$$\mathrm{R}(q,d_j) = \frac{P(\vec{d}_j|R)}{P(\vec{d}_j|\bar{R})}$$

- Suponiendo los términos índice independientes, podemos factorizar la probabilidad un producto para cada término:

$$\mathrm{R}(q,d_j) = \frac{\left(\prod_{g_i(\vec{d}_j=1)} P(k_i|R)\right)\cdot\left(\prod_{g_i(\vec{d}_j)=0} P(\bar{k_i}|R)\right)}{\left(\prod_{g_i(\vec{d}_j)=1} P(k_i|R)\right)\cdot\left(\prod_{g_i(\vec{d}_j)=0} P(k_i|R)\right)}$$

donde $P(k_i|R)$ es la probabilidad de que un término aparezca en un documento aleatoriamente seleccionado de R, y g_i es una función que extrae el peso de un término i del documento.

- Tomando logaritmos y aplicando que $P(k_i|R) + P(k_i|R) = 1$ queda

$$\mathrm{R}(q,d_j) = \sum_{i=1}^{T}\left\{w_{i,q}\cdot w_{i,j}\cdot\left(\log\frac{P(k_i|R)}{1-P(k_i|R)} + \log\frac{1-P(k_i|\bar{R})}{P(k_i|\bar{R})}\right)\right\}$$

2.4.- El modelo probabilístico básico (3)

- *Problema final:* ¿Cómo obtener cada uno de los $P(k_i|R)$, si R es desconocido?.
- Vamos a hacer la recuperación en dos pasos.

En el primero, obtendremos un estimador de *R.* Para ello, necesitamos hacer **dos suposiciones:** (ausencia total de conocimiento).

1. $P(k_j|R)$ es constante para todos los términos (0.5, por ejemplo)
2. La distribución de términos entre los documentos no relevantes es la misma que la distribución de los términos entre todos los documentos.

- Es decir:

$$P(k_i|R) = 0.5$$

$$P(k_i|\bar{R}) = \frac{n_i}{N}$$

- Una vez recuperados los documentos, tomamos este conjunto como conjunto V (sólo tomamos los n primeros si V es grande). Sea V_i el subconjunto de documentos de V que contienen al término i.

2.4.- El modelo probabilístico básico (4)

- Una vez que contamos con *V,* éste estimará a *R*, con lo que podemos recalcular las probabilidades $P(k_j|R)$ y $P(k_j|\neg R)$ (ya no utilizamos las suposiciones anteriores). Realizamos dos nuevas suposiciones:

1. Podemos aproximar $P(k_j|R)$ a la distribución de los términos k_j entre los documentos recuperados, es decir $P(k_j|R) = |V_j|/|V|$

2. Podemos aproximar $P(k_j|\neg R)$ suponiendo que los documentos no recuperados no son relevantes, es decir $P(k_j|\neg R) = (n_j - |V_j|)/(N-|V|)$

Recalculando las probabilidades de esta forma, se pueden recalcular los valores finales de relevancia de cada documento según la fórmula anterior.

Para evitar casos límite (valores muy pequeños de $|V|$, como 1 ó 0), también podemos definir las fórmulas $P(k_j|R) = (|V_j| + 0.5)/(|V| + 1.0)$

$P(k_j|\neg R) = (n_j - |V_j| + 0.5)/(N-|V| + 1.0)$

2.4.- El modelo probabilístico básico (5)

Análisis Crítico


Recuperación “ordenada”.


Acoplamiento “parcial”.


Soportado por una teoría formal.


Peores resultados que el modelo de espacio vectorial.


Difícil de entender. Muchas suposiciones.

3.1.- Introducción a la implementación de un SRI

- Una colección de documentos (p. ej. páginas web) no está preparada para realizar directamente operaciones de R.I. Por ejemplo: *encontrar la lista de páginas que contengan los términos "Fernando" y "Alonso".*

- No es lógico pensar que la búsqueda se haga linealmente. Las colecciones actuales de documentos tienen Gigas (¡o Teras!). Con las tasas de transferencia actuales, despreciando el tiempo de cómputo, y suponiendo que los datos son secuenciales, en una colección de 10 Gb habría que hacer varias búsquedas de 10 segundos (y en la realidad sería bastante más).

- Necesitamos crear estructuras **persistentes** que permitan acceder eficientemente a los datos ya procesados.

3.2.- Estructuras de datos: el índice invertido

- Similar al índice terminológico de un libro: para cada término, tenemos la lista de (páginas) identificadores de documentos, en los que aparece. También puede dar la lista de pesos de cada término en cada documento.

- Se construye una sóla vez, durante un proceso llamado ***indexación.***

- Obviamente, el índice no cabe en memoria (tiene un tamaño del mismo orden que el de la colección), por lo que se divide en dos partes:
  - Vocabulario (lexicon): que contiene la lista de términos
  - Ocurrencias (occurrences): que es una tabla con las listas de documentos (y los pesos).

- Cada entrada en el vocabulario contiene la dirección en disco donde se encuentra su ocurrencia. Así, el vocabulario se puede mantener en memoria (10 MB por cada 300 MB de colección)

3.2.- Estructuras de datos: el índice invertido (2)

- La construcción del índice, al no caber en memoria, no es sencilla. Normalmente, se realiza en dos pasos:
  1. Obtención (a disco) de las tuplas *(t, d, f(t,d), w(t,d))* (todas con el mismo *d* para cada documento). Estarán ordenados por *d*.
  2. Utilizando un algoritmo de ordenación externa, ordenar por *t*.
  3. Una vez ordenado el archivo de tuplas, ir leyendo todas las tuplas de un mismo *t* y crear la ocurrencia correspondiente.

- Obviamente, se puede añadir información sobre la posición de las palabras en el texto (para hacer consultas por proximidad; con "comillas").

- Por otra parte, el vocabulario debe permitir búsqueda *O(1)* por identificador, y *O(log n),* por cadena (*n* el número de términos).

- Una buena referencia para los algoritmos de indexación (y en general, las estructuras de datos es el libro *Managing Gigabytes*).

3.3.- Implementación de un sistema basado en E.V.

- La implementación del algoritmo de indexación es común para todos (salvo detalles). Lo que especificamos es el de recuperación:

Algoritmo:

1. A = {} (acumuladores para cada documento)
2. Para cada término *t* de la consulta
   - Obtener la dirección de su ocurrencia L(t)
   - Leer la ocurrencia L(t) de disco
   - Para cada par $<d, w_{d,t}>$ en L(t)
     - Si $A_d \notin A$, iniciar A_d a 0 y añadirlo al conjunto A
     - $A_d = A_d + w_{d,t}$
3. Para cada A_d de A, normalizar $A_d = A_d/W_d$
4. Devolver al usuario los *r* mejores documentos

Bibliografía

General

R. Baeza-Yates, B. Ribeiro-Neto, **Modern Information Retrieval**, Addison Wesley Longman, 2002.

C. J. van Rijsbergen, **Information Retrieval, (Second edition)**, Butter Worths, London, 1979. (GRATIS EN INTERNET!!!)

Modelo de espacio vectorial

G. Salton , A. Wong , C. S. Yang, A vector space model for automatic indexing, Communications of the ACM, v.18 n.11, p.613-620, Nov. 1975

Modelos probabilísticos

F. Crestani, M. Lalmas, C. J. van Rijsbergen, I. Campbell, *"Is This Document Relevant? ... Probably": A Survey of Probabilistic Models in Information Retrieval*, ACM Comput. Surv. **30**(4): 528–552, 1998.

Estructuras de datos e implementación

I. H. Witten, A. Moffat, T. C. Bell **Managing Gigabytes**, Morgan Kaufmann, 1999.

Gracias
por vuestra atención